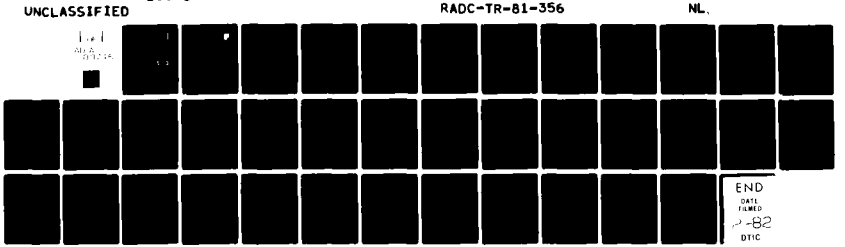
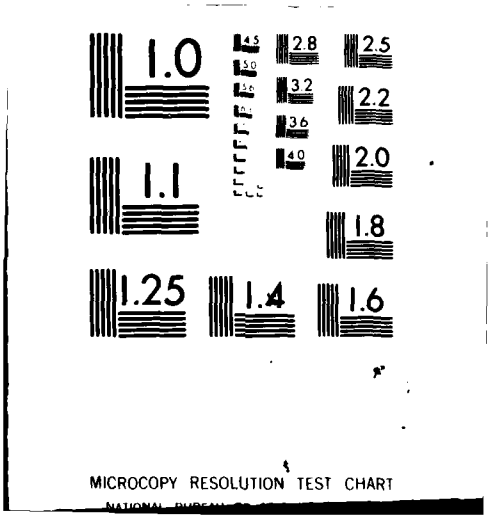
AD-A109 745 INTERMETRICS INC CAMBRIDGE MA F/G 9/2
ADA INTEGRATED ENVIRONMENT I. SYSTEM SPECIFICATION.(U)
DEC 81 F30602-80-C-0291
UNCLASSIFIED RADC-TR-81-356 NL



END
DATE FILMED
2-82
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
MICROCOPY RESOLUTION TEST CHART

PHOTOGRAPH THIS SHEET

AD A109745

DTIC ACCESSION NUMBER

LEVEL

INVENTORY

Intermetrics, Inc.
Cambridge, MA

ADA Integrated Environment I System Specification

DOCUMENT IDENTIFICATION

RADC-TR-81-356

Contract F30602-80-C-0291

Dec. 81

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

DISTRIBUTION STATEMENT

ACCESSION FOR	
NTIS	GRA&I
DTIC	TAB
UNANNOUNCED	
JUSTIFICATION	
BY	
DISTRIBUTION /	
AVAILABILITY CODES	
DIST	AVAIL AND/OR SPECIAL
A	

DISTRIBUTION STAMP

DTIC
ELECTE
S JAN 19 1982 D
D

DATE ACCESSIONED

82 01 12 019

DATE RECEIVED IN DTIC

PHOTOGRAPH THIS SHEET AND RETURN TO DTIC-DDA-2

DTIC FORM OCT 79 70A

DOCUMENT PROCESSING SHEET

RADC-TR-81-356
Interim Report
December 1981

ADA INTEGRATED ENVIRONMENT I SYSTEM SPECIFICATION

Intermetrics, Inc.

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

This document was produced under Contract F30602-80-C-0291 for the Rome Air Development Center. Mr. Don Roberts is the COTR for the Air Force. Dr. Fred H. Martin is Project Manager for Intermetrics.

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TR-81-356 has been reviewed and is approved for publication.

APPROVED: Donald F. Roberts

DONALD F. ROBERTS
Project Engineer

APPROVED: John J. Marciniak

JOHN J. MARCINIAK, Colonel, USAF
Chief, Command and Control Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief, Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (COES) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document requires that it be returned.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER RADC-TR-81-356	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) ADA INTEGRATED ENVIRONMENT I SYSTEM SPECIFICATION		5. TYPE OF REPORT & PERIOD COVERED Interim Report 15 Sep 80 - 15 Mar 81
		6. PERFORMING ORG. REPORT NUMBER N/A
7. AUTHOR(s)		8. CONTRACT OR GRANT NUMBER(s) F30602-80-C-0291
9. PERFORMING ORGANIZATION NAME AND ADDRESS Intermetrics, Inc. 733 Concord Avenue Cambridge MA 02138		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62204F/62702F/33126F 55811908
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (COES) Griffiss AFB NY 13441		12. REPORT DATE December 1981 13. NUMBER OF PAGES 31
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Same		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Donald F. Roberts (COES)

Subcontractor is Massachusetts Computer Assoc.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Ada	MAPSE	AIE
Compiler	Kernel	Integrated environment
Database	Debugger	Editor
KAPSE	APSE	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The Ada Integrated Environment (AIE) consists of a set of software tools intended to support design, development and maintenance of embedded computer software. A significant portion of an AIE includes software systems and tools residing and executing on a host computer (or set of computers). This set is known as an Ada Programming Support Environment (APSE). This system specification describes the basic design for a minimal APSE, called a MAPSE. The MAPSE is the foundation upon which an

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

APSE is built and will provide comprehensive support throughout the design, development and maintenance of Ada software. The MAPSE tools described in this specification include an Ada compiler, linker/loader, debugger, editor, and configuration management tools. The kernel (KAPSE) will provide the interfaces (user, host, tool), database support, and facilities for executing Ada programs (runtime support system).

TABLE OF CONTENTS

INTERMETRICS INCORPORATED • 733 CONCORD AVENUE • CAMBRIDGE, MASSACHUSETTS 02138 • (617) 661-1840

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF FIGURES AND ILLUSTRATIONS

INTERMETRICS INCORPORATED • 733 CONCORD AVENUE • CAMBRIDGE, MASSACHUSETTS 02138 • (617) 661-1840

1.0 SCOPE

This specification establishes the performance, design, development and test requirements for the Ada Integrated Environment (AIE).

INTERMETRICS INCORPORATED • 733 CONCORD AVENUE • CAMBRIDGE, MASSACHUSETTS 02138 • (617) 661-1840

2.0 APPLICABLE DOCUMENTS

Please note that the bracketed number preceding the document identification is used for reference purposes within the document.

2.1 Government Documents

[G-1] Requirements for Ada Programming Support Environments, "STONEMAN", February 1980, Dept. of Defense.

[G-2] Reference Manual for the Ada Programming Language, proposed standard document, U.S. Department of Defense, July 1980.

[G-3] Formal Definition of the Ada Programming Language, Preliminary Version for Public Review, November 1980, U.S. Government as represented by the Director, Defense Advanced Research Projects Agency.

2.2 Non-Government Documents

[N-1] IBM 370/VM Specifications

[N-2] Perkin/Elmer 8/32 Specification

[N-3] Diana Reference Manual, G. Goos and Wm. Wulf, Institut Fuer Automatik II, Universitaet Karlsruhe and Computer Science Dept., Carnegie-Mellon University, March 1981.

[N-4] IBM Virtual Machine Facility/370: System Programmer Guide, IBM, Inc.

[N-5] OS/32 Programmer Reference Manual, Perkin-Elmer, Computer Systems Division, Oceanport, N.J., April 1979.

[I-1] Intermetrics LG System Description, 31 August, 1980, IR-536.

[I-2] LG User's Guide, December 1979, IR-427.

Computer Program Development Specifications for Ada Integrated Environment (Type B5), March 1981:

[I-3] Ada Compiler Phases, IR-677.

[I-4] KAPSE/Database, IR-678.

[I-5] MAPSE Command Processor, IR-679.

[I-6] MAPSE Generation and Support, IR-680.

[I-7] Program Integration Facilities, IR-681.

[I-8] MAPSE Debugging Facilities, IR-682.

[I-9] MAPSE Text Editor, IR-683.

[I-10] Technical Report (Interim), IR-684.

3.0 REQUIREMENTS

3.1 System Definition

The Ada Integrated Environment (AIE) is an integrated set of software (and hardware) tools intended to support the life cycle development and maintenance of embedded computer software, written principally in the Ada programming language. A significant portion of an AIE is composed of those software systems and tools residing and executing on a host computer (or set of computers constituting a host); such a set is termed an Ada Programming Support Environment (or APSE). This System Specification describes the foundations upon which an APSE can be built; that is, a Minimal Ada Programming Support Environment (or MAPSE). It should be noted that the MAPSE is not just a stepping stone toward the development of an APSE, but itself constitutes a usable and useful programming environment. Although minimal, the MAPSE provides comprehensive support throughout the development and maintenance of Ada software. Unless otherwise noted, all specifications and paragraphs below pertain to a MAPSE.

3.1.1 General Description

The MAPSE is composed of seven functional areas (facilities or tools): (1) KAPSE/Database; (2) Command Processor; (3) Program Integration; (4) Ada Compiler Phases; ((5) Text Editor; (6) Debugger, and (7) Generation and Support.

3.1.1.1 KAPSE/Database

The Kernel Ada Programming Support Environment (or KAPSE) provides a machine-independent portability interface as well as database communications, program control, and run-time support functions. The KAPSE facilitates portability by isolating dependencies and by providing functional interfaces and capabilities to software tools and users.

In addition, the KAPSE/Database design includes built-in support for a variety of configuration management functions, providing the user with an efficient means of identifying, collecting, and/or associating groups of objects in the database.

3.1.1.2 Command Processor

The Command Processor is the primary MAPSE tool by which the user gains access to other MAPSE tools and capabilities. The Command Processor consists of: (1) a user interface processor, which accepts and analyzes user commands expressed in the MAPSE command language (MCL); and (2) a system command executive, which activates, suspends, and/or terminates execution of MAPSE tools and other Ada programs via KAPSE program-control facilities.

3.1.1.3 Program Integration

Program Integration is accomplished by the two major MAPSE tools that deal with the Ada Program Library: the Ada Compiler, and the linker. The Ada Compiler controls the translation of Ada source text into linkable units. The linker combines the units into executable programs. Program Integration also includes additional tools relating to the Ada program library, such as a body (stub) generator, status report processor, and tools to control automatic recompilation.

3.1.1.4 Ada Compiler Phases

The actual translation of Ada programs is carried out by distinct phases of the compiler. The compiler phases are grouped into three sections: (1) the front end, which performs syntactic and semantic analyses to create a standard intermediate language form (called Diana); (2) the middle section, which transforms the Diana representation to a lower level that exposes the run-time model and performs Ada-specific optimizations; and (3) the back end, which generates optimized target code in linkable form for the MAPSE host.

3.1.1.5 Text Editor

The Text Editor is a MAPSE tool for manipulating general text, such as Ada program source code, documentation, and user command scripts.

3.1.1.6 Debugging Facilities

Debug is a MAPSE tool that controls the progress of an executing Ada program through the definition and subsequent encounter of execution control points (breakpoints). At such points, a variety of debugging actions may be defined. Such actions may request information about program objects or may alter the flow of control to aid in the debugging of Ada programs.

Debug consists of: (1) an input processor, which analyzes user-generated debugging commands; and (2) a debug executive, which responds at breakpoints and performs the user-specified actions. Interfaces are defined within Debug to allow the addition of comprehensive APSE tools supporting advanced simulation/emulation techniques.

3.1.1.7 MAPSE Generation and Support

The MAPSE Generation and Support facility is a collection of programs, routines, and/or data that support the construction and maintenance of the MAPSE itself. These include: (1) the bootstrap compiler; (2) the parser and lexer generators; (3) the Virtual Memory Management (VMM) System; and (4) routines to facilitate rehost of the MAPSE. These facilities are designed to maintain the MAPSE on the MAPSE itself.

3.1.2 Mission

The mission, or objective, of the MAPSE is to provide a minimal APSE that can serve both as an open-ended prototype and as a working environment consisting of tools and aids to assist programmers and project managers in the development and maintenance of Ada software. To the greatest extent possible, the MAPSE shall be rehostable and retargetable to a variety of suitable host computers.

3.1.3 Threat (N/A)

3.1.4 System Diagrams

The top level functional flow diagram for the MAPSE is shown in Figure 1.

3.1.5 Interface Definition

3.1.5.1 System Interface Definition

The major MAPSE interfaces with other systems are concentrated within the KAPSE. These include interfaces to the bare host machines (if applicable), to the CP-VM370 operating system (IBM 370/VM), and to the OS/32 operating system (Perkin-Elmer 8/32).

(a) MAPSE on a Bare Machine. The KAPSE includes interfaces to the bare machine resources (processors, core storage, I/O lines, etc.), to the raw storage devices (discs, tapes, etc.), to keyboard or other terminal I/O devices for the purpose of data transfer from/to users or other machines, and an import/export function for data transfer to/from software systems outside the MAPSE.

(b) MAPSE on an IBM 370/VM. The KAPSE includes those interfaces listed above (paragraph (a)) plus interfaces to the CP-VM370 operating system (see 2.2).

(c) MAPSE on a Perkin-Elmer 8/32. The KAPSE includes those interfaces listed in paragraph (a) plus interfaces to the OS/32 operating system (see 2.2).

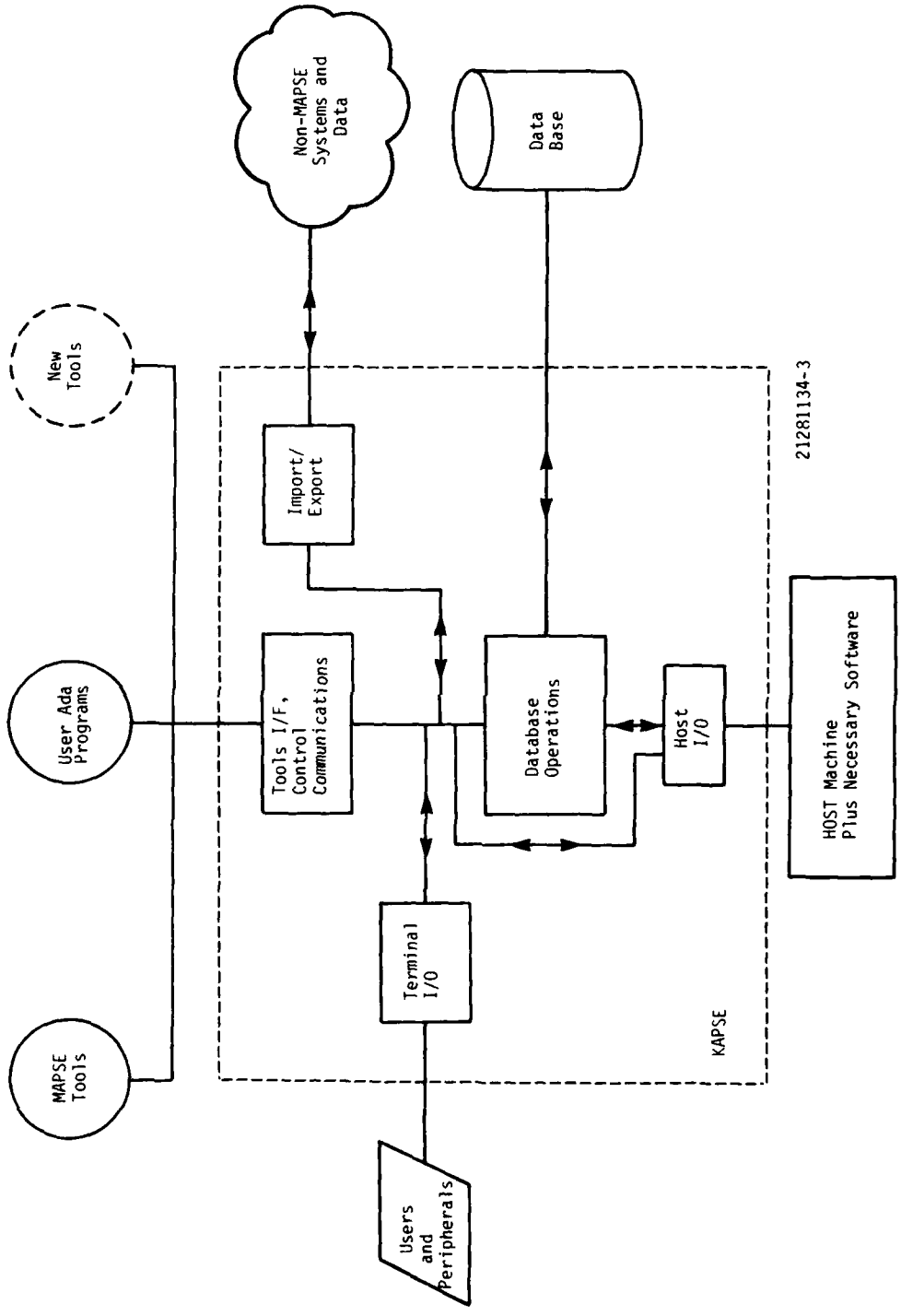


FIGURE 1: MAPSE OVERVIEW

3.1.5.2 Interfaces Within the MAPSE.

In general, the functional areas described under Section 3.1.1 (with the exception of paragraph 3.1.1.7) interface with the KAPSE (i.e., KAPSE functions and routines) and only the KAPSE. These interfaces take either of two forms - control or data. Control interfaces define capabilities by which one tool can affect another. Such capabilities may be viewed as requests that are implemented by the program-control routines within the KAPSE. Data interfaces define collections of data, or records, which are interpreted by one or more tools. The database serves as a repository for all data. The Data Base Manager, within the KAPSE, controls and monitors all requests for reading, writing, storing and retrieving data. The KAPSE also contains the Loader, which accepts as input a "nearly" executable module. As output, the Loader produces an executable module and loads it into storage ready for execution.

(a) Control Interfaces. The KAPSE provides facilities to support the control modes given in Table 1, below, among executing Ada programs. (Each mode may include a set of data and/or control parameters.)

(b) Data Interfaces. The interfaces shown in Table 2 are defined for each tool. All data resides within the database and is accessed via KAPSE database operations.

Mode	Description
Unrelated	Separate programs unsynchronized except by database operations controlling common data base access requests
Invoker/ Invoked	Invoking program is suspended until invoked program completes. (This is most common and equivalent to a procedure call.)
Creator/ Created	Creator may continue after creating (e.g., background execution, or the command processor starting a program).
Controller/ Controlled	Controller may start, stop, or modify controlled program (e.g., debugger taking action at a control point).
Requestor/ Servicer	Servicer exists independent of multiple requestors (e.g., the KAPSE database operations).

TABLE 1: KAPSE Control Modes

Tool	Interfaces
Ada Compiler (may be requested by Command Processor, Program Integration, or an executing Ada program.)	Input: Text, forms of Diana Output: Various forms of Diana [N-3] listings
Command Processor (activated automatically by logging onto the MAPSE; may also be requested by an executing Ada program.)	Input: Text Output: Control parameters and requests
Text Editor (may be requested by the Command Processor or an executing Ada program)	Input: Text, scripts Output: Text, scripts
Debug (may be requested by the Command Processor or an executing Ada program.) The execution of an Ada program may be controlled by Debug.	Input: Text, scripts, various forms of Diana Output: Control parameters and requests
Program Integration (may be requested by the Command Processor or an executing Ada program)	Input: Text, scripts, various forms of Diana, including linkable Diana Output: Text, executable programs control parameters and requests
MAPSE Generation & Support (may be requested by Command Processor or an executing Ada program)	Input: Text, binary and ASCII forms of data structures (including Diana as a subset) Output: Text, binary and ASCII forms of data structures
KAPSE functions (may be requested by Command Processor, Program Integration, executing Ada program, or host system)	Input: Requests, text scripts, database objects, MAPSE-external data Output: Database objects (including executables), MAPSE external data.

TABLE 2: Generic Data Interfaces

3.1.6 Government Furnished Property List

3.1.6.1 Perkin-Elmer Equipment

(a) Disk drives (2)
Company: Wangco Inc.

Model #ST-2222
Serial #46057

1. Fixed disc 5 M Bytes

2. Removable disk 5 M Bytes

(b) Mag Tape Drives (2)
Company: Wangco Inc.

Model #1045
Serial #12061, 12037

45 ips, 800 cpi

(c) Card Reader
Company: Documation

Model #M1000L
Serial #7708976PR

1000 cpm

(d) Printer
Company: Data Printer Corp.

Model #V-132-C
Serial #41624070

300 LPM

(e) 8132 Model w/OS 8/32-REV3 Now
OS 8/32-REV6 Future

Serial #2580
w 300 Nan0 Sec response
128K memory with 256 K bytes to be added
Host terminal

FOX 1100 TERMINAL
Serial #100501

(f) Interactive Terminals (TBD)

(g) All necessary defining and operating manuals

3.1.6.2 IBM Equipment (TBD)

3.1.7 Operational and Organizational Concepts

The MAPSE shall be installed on an IBM 370/VM and on a Perkin/Elmer 8/32 computer, both at Rome Air Development Center.

3.2 Characteristics

3.2.1 Performance Characteristics

The MAPSE shall provide a minimal programming support environment to aid programmers and managers in the development and maintenance of Ada computer programs. It shall be designed to run on at least two host computers; viz the IBM 370/VM and the Perkin/Elmer 8/32. For the IBM host, the MAPSE shall be compatible with the CP-370/VM operating system; for the Perkin/Elmer, the MAPSE shall be compatible with the OS-32 operating system. The MAPSE shall support the following operational modes and functions:

1. compile, link, load, execute, and debug programs written in the full Ada programming language on the specified host machines. Debugging shall be accomplished in terms of Ada statements, symbols, names, etc.;

2. build and develop (and maintain) Ada programs by linking (and maintaining) collections of separate Ada compilations within appropriate program libraries. Such programs may be general Ada user programs, MAPSE tools, KAPSE functions, new APSE tools, and embedded computer software. Embedded computer software may be intended to execute on the host or intended for execution on a target machine, but developed in the host environment;

3. process a user command language to activate and otherwise control (job control) all of the MAPSE tools and functions (See 3.1) in batch, remote-batch, and on-line modes. Allow such control to be exercised by both users and executing Ada programs;

4. accept and manipulate (edit) user-supplied text representing Ada source programs, program documentation, and other general text. User interfaces shall be supported by batch, keyboard interactive, and high-band-width graphics terminals. The standard Ada character set is specified;

5. establish and maintain (that is, enter, retrieve, manipulate) a central data base as a respository of source text, derived results (from compilation and execution), and other documentation;

6. provide an ability to organize and view the database as an hierarchical or relational set of objects. Each object shall belong to a category and possess distinguishing characteristics by which, in combination, it can be uniquely referenced and used. In addition, a history shall be appended to each object so that it can be re-established if necessary;

7. provide mechanisms for establishing a variety of management disciplines and protocols in conducting a project to develop Ada software. Mechanisms shall be based on controlling access to database objects (including tools) by a system of allocated and enforced permissions;

8. establish and maintain configuration control by associating objects into controlled configurations and arbitrary partitions, based on their distinguishing attributes;

9. support simultaneous use of the system by more than one user, either in cooperation or acting independently (that is, in a non-interfering manner);

10. when operating on the IBM 370 under the CP-VM370 operating system or on the Perkin/Elmer 8/32 under OS-32, the MAPSE shall not preclude use of the machine by non-MAPSE users. In such cases, MAPSE data, tools, and functions shall be fully protected from interference. A MAPSE/non-MAPSE interface shall be supported via an import/export function.

11. design tools and supporting functions so that the MAPSE can be maintained on the MAPSE itself.

3.2.1.1 Operational Characteristics

Figure 3 is a diagram illustrating the MAPSE operational characteristics. Circled numbers indicate significant actions or operational interfaces and are defined as follows:

User View of the MAPSE

1. User logs on and enters comands or text.

2. Command Processor is activated automatically when user logs on.

3. Command Processor can activate any MAPSE tool or function through the KAPSE program control and loader routines.

4. Desired Ada program is loaded for execution.

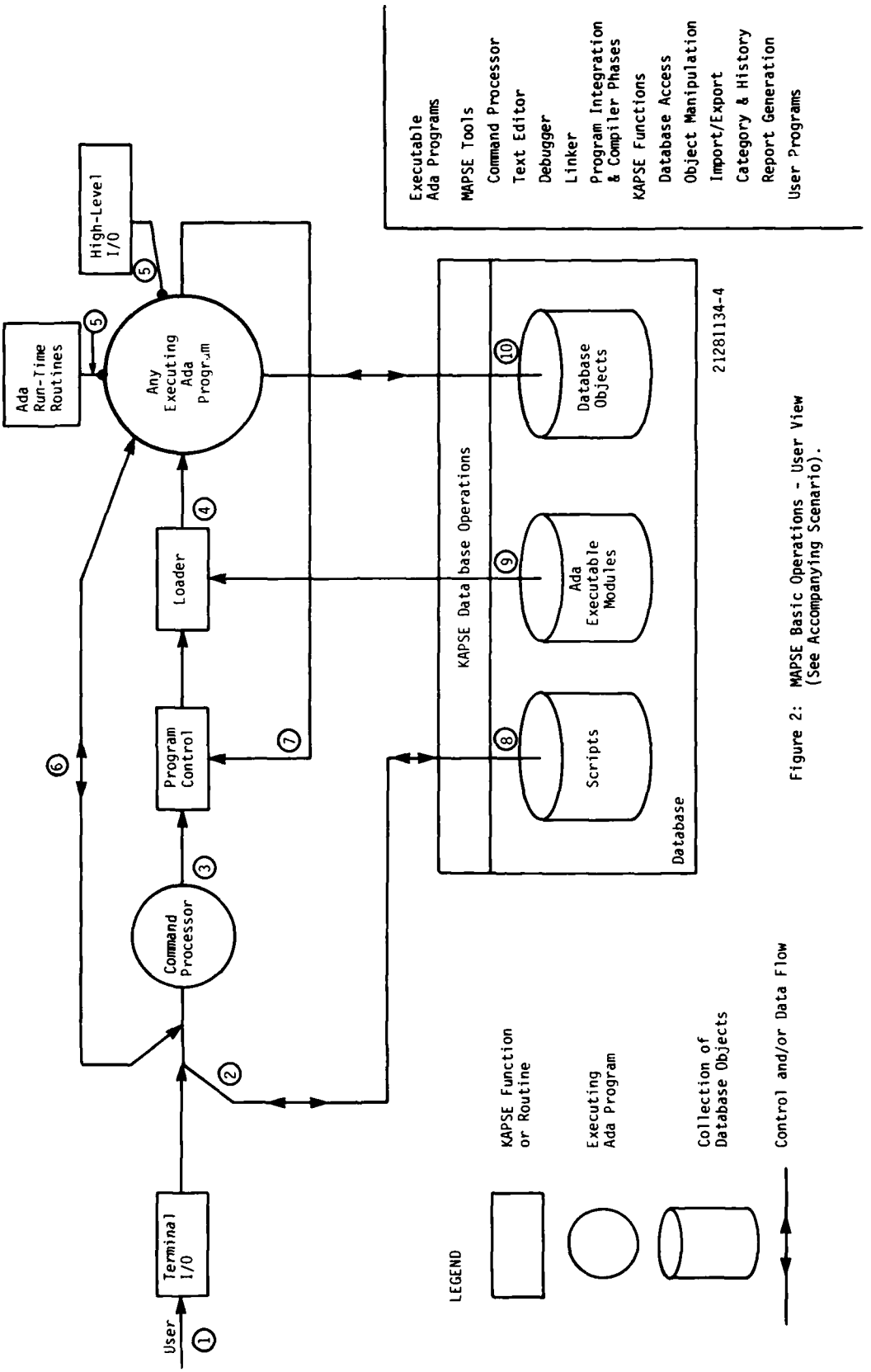


Figure 2: MAPSE Basic Operations - User View (See Accompanying Scenario).

FIGURE 2: MAPSE Basic Operations - User View
(See Accompanying Scenario)

5. Execution is supported by Ada run-time and high level I/O routines within KAPSE.

6. User can interface to executing Ada program.

7. Executing Ada program can activate any MAPSE tool or function (including Command Processor) through the KAPSE program control and loader routines.

8. Stored scripts within the database (created via the Editor), can affect the Command Processor or any executing Ada program in the same manner as the user.

9. Stored Ada executable modules within an Ada Program Library in the database (created via the Linker) are accessed by KAPSE database operations and loaded for execution by the KAPSE loader.

10. General database objects are stored within, and retrieved from the database by executing Ada programs using the KAPSE database operations.

3.2.1.2 Design Goals

1. Usability - the system shall be easy to use for novice as well as experienced users. Simple things shall be simple to do without full knowledge or exercise of all options and capabilites.

2. Applicability - The MAPSE shall be applicable to the development and maintenance of Ada programs, including the MAPSE itself, as well as embedded computer software for DoD projects: large and small, of varied duration, and subject to a variety of management requirements and disciplines. While providing specific tools, the MAPSE shall not dictate the way in which they are used. The system shall provide support only; the programmers will program and the managers will manage.

3. Portability - the system shall be portable, in that it can be rehosted and retargeted to other host computers with minimum effort. To support this requirement, host dependencies shall be isolated so that they are easily identified. In addition, maximum use shall be made of the Ada programming language in implementing the MAPSE.

4. Functionality - rehosted MAPSE's shall exhibit constant (in so far as is possible) functionality.

5. Modularity - the MAPSE shall be constructed in a modular fashion so that functional areas (tools) are lightly coupled and can easily be replaced, modified, or extended.

6. Open-Endedness - the MAPSE shall be an open-ended system so that new tools and facilities can be added within the context of the original design. The MAPSE shall support and be the foundation for the development and implementation of a full APSE and AIE.

7. Integrated Design - an integrated approach shall be taken to the design of the MAPSE, making maximum use of uniform interfaces and reusable portions.

8. Reliability - the reliability of the MAPSE shall be ensured by the use of dependable, disciplined design and development methodologies; it shall be designed for testability (with built-in test techniques), and full qualification shall be undertaken (PQT, FQT) prior to release for use. The system shall contain features to protect itself from user and system errors. Built-in aids shall be provided to detect anomalies in operation and to correct such anomalies with a minimum of effort. These aids shall be compatible with the use of Ada and modular construction techniques.

3.2.2 Physical Characteristics

Physical characteristics shall be as specified for equipment designated in Section 3.1.6. No particular security criteria shall be applicable to the MAPSE.

3.2.3 Reliability

Reliability of hardware (computer) facilities shall be as specified for equipment designated in Section 3.1.6. Reliability of the MAPSE is discussed in Section 3.2.1.2(8).

3.2.4 Maintainability

3.2.4.1 Equipments and Vendor Supplied Support Software

Maintainability shall be as prescribed in the specifications listed in 3.1.6.

3.2.4.2 MAPSE Software Systems

The MAPSE shall be maintained on both hosts by contractor personnel in accordance with a Maintenance Manual for at least 12 months following delivery and acceptance of the MAPSE by the Air Force. Thereafter, maintenance procedures shall be specified in sufficient detail to permit maintenance of the system by other than

the designing contractor. Maintenance shall consist of the corrections of operational and documentation errors experienced during test or field use, and necessary modifications to meet specification changes. (See 3.2.1.2(8)).

3.2.5 Availability

Following acceptance by the Air Force, the MAPSE shall be in an operable and maintainable state on both the IBM 370/VM and Perkin/Elmer computers located at RADC. Use, schedule, and commitment of the system shall be by Air Force/RADC direction.

3.2.6 System Effectiveness Models (N/A)

3.2.7 Environment Conditions

Environment conditions shall be as specified in Section 3.1.6.

3.2.8 Nuclear Control Requirements (N/A)

3.2.9 Transportability

The MAPSE shall be transportable with minimum effort from the IBM 370/VM host compiler to the Perkin/Elmer 8/32 host computer. In addition, the MAPSE shall be designed for transportability to any other host computer (or network of computers) that can reasonably be anticipated at the time of design. Such computers shall include all appropriate 32 bit (or greater) mainframes of general register or stack architecture, but need not include state-of-the-art machines, (e.g., pipeline, array and parallel processing architectures). The design of the MAPSE/KAPSE shall facilitate transportability by isolating all machine dependencies within the KAPSE, by implementing the MAPSE/KAPSE (to the greatest degree possible) in the Ada programming language, and by guaranteeing constant user functionality regardless of host.

3.3 Design and Construction

The MAPSE shall be designed and constructed in accordance with the methods put forth in the Computer Program Development Plan.

3.3.1 Materials, Processes and Parts

Materials, processes, and parts selection shall be as specified in Section 3.1.6.

3.3.2 Electromagnetic Radiation

System components shall be electromagnetically compatible with themselves and associated equipments as specified in Section 3.1.6.

3.3.3 Nameplates and Product Marking

Nameplates and Product Marking shall be as specified in Section 3.1.6.

3.3.4 Workmanship

3.3.4.1 Hardware

As specified in Section 3.1.6

3.3.4.2 MAPSE

In accordance with the Computer Program Development Plan.

3.3.5 Interchangeability

3.3.5.1 Hardware

As specified in Section 3.1.6

3.3.5.2 MAPSE

MAPSE software shall be modularly designed with well-defined interfaces and utilizing Ada's abstract data type facilitates to a maximum degree. Specifically with respect to the Ada Compiler, the back end (or code generating portion) shall be replaceable, meeting a well defined Diana intermediate language interface specification [N-3].

3.3.6 Safety

Safety shall be as specified in Section 3.1.6.

3.3.7 Human Performance/Human Engineering

3.3.7.1 Hardware

The design or addition of new equipment requiring man/machine interfaces shall be as specified in Section 3.1.6.

3.3.7.2 MAPSE

The MAPSE system design shall include features to protect itself from user and system errors. (See 3.2.1.2(8)). A User's Manual shall be provided to guide personnel in the use of MAPSE (See 3.4)

3.3.8 Computer Programming

Computer programming to implement the MAPSE shall be done in Ada unless prior government approval is obtained and in accordance with the Computer Program Development Plan.

3.4 Documentation

The following documents shall be provided to the MAPSE User's community:

1. MAPSE User's Manual
2. MAPSE Maintenance Manual
3. MAPSE Rehostability Procedure Manual
4. MAPSE Retargetability Guidelines Manual

These documents shall be produced and specified in accordance with the following standards and specifications (listed in Section 2.)

AIE, System Specification (this document)

Computer Program Development Specifications for each functional area identified in the System Specifications (See 3.1)

Computer Program Product Specifications for each Computer Program Configuration Item (CPCI) derived from the set of Development Specifications

Test Plans for each CPCI

3.5 Logistics

3.5.1 Maintenance

3.5.1.1 Hardware

Hardware maintenance shall be in accordance with Section 3.1.6.

3.5.1.2 MAPSE

Maintenance of the MAPSE system shall be accomplished in accordance with the Maintenance Manual (see 3.4)

3.5.2 Supply (N/A)

3.5.3 Facilities and Facility Equipment

Facilities and facilities equipment shall be as specified in Section 3.1.6.

3.6 Personnel and Training

3.6.1 Personnel

The MAPSE shall be designed to be used by both novice and experienced personnel engaged in the management and/or development of Ada software. Operations on each host shall be conducted, respectively, by personnel familiar with the IBM 370/VM computer and it's CP-VM370 operating system and with the Perkin/Elmer 8/32 computer and its operating system.

3.6.2 Training

3.6.2.1 Operations Training

A one week training course in the use and operation of the MAPSE shall be conducted at RADC for up to 20 personnel soon after acceptance of the system by the Air Force.

3.6.2.2 Maintenance Training

A two week training course in the maintenance, rehosting and retargeting of the MAPSE shall be conducted at RADC for up to ten Ada programming personnel.

3.7 Functional Area Characteristics

The MAPSE is subdivided into the functional areas listed in 3.1.1; they shall exhibit the characteristics described below.

3.7.1. KAPSE/Database

The KAPSE/Database shall provide all the external and internal interfaces associated with the MAPSE and include the operations necessary to manipulate, organize, and control access to the database. The database shall be the repository of all data objects and their attributes including source text, the various forms of Diana, executable modules, intermediate results of program execution, and any other data defined and associated with MAPSE tools or operations.

More specifically, the KAPSE/Database shall:

1. provide all the interfaces between the MAPSE and any software or hardware systems outside the MAPSE;

2. encapsulate all hardware and host dependencies, including interfaces to vendor-supplied support software;

3. provide the run-time routines to effect an Ada virtual machine;

4. provide keyboard/terminal interfaces to users and/or other equipments;

5. include database operations to assure exclusive access to and control of the database including management and summary reporting of: configurations, partitions, attributes, archiving and protection;

6. provide standard data and program control interfaces among all defined MAPSE tools and the database, and supply uniform interface definitions for future tools;

7. facilitate invocation of any tool from any other tool, including the Command Processor;

8. include a loader to load the host memory with executable units from an Ada program library;

9. provide high-level Input/Output capabilities similar to those available with FORTRAN;

10. provide the necessary capabilities to effect system recovery/reconfiguration after experiencing a malfunction.

3.7.2 Command Processor

The Command Processor shall be the primary interface between the user and the MAPSE tools and facilities. The Command Processor shall be activated automatically when a user logs into the MAPSE, or when requested by an executing Ada program.

More specifically, the Command Processor shall implement a MAPSE Command Language (MCL) with which the user can perform the following functions:

1. invoke and control the background execution of an arbitrary Ada program;

2. receive data from a program, create and manipulate variables and query the state of a program;

3. request help on a per-program basis;

4. connect arbitrary sequences of MCL commands for execution in foreground or background;

5. redirect the standard input or output of MCL sequences from/to arbitrary files.

3.7.3 Program Integration

MAPSE Program Integration facilities shall control all access to, and manipulation of, the Ada program libraries, including compilation and linkage editing. Since the libraries are proper objects of the database, access must be gained through the KAPSE database operations (See 3.1.1.1, 3.2.1.1). More specifically, Program Integration shall:

1. control the compilation process by directing new Ada source or stored intermediate language forms of Ada source (from the program library) to the several Ada compiler phases for compilation and/or recompilation;

2. minimize the requirement for recompilations by analyzing the interdependencies of the unit(s) presented for compilation with previously compiled (and stored) library units, and recompile only when necessary;

3. provide a compiler driver that performs an analysis to determine which library units must be recompiled to maintain a consistent compilation set;

4. support a program build by selecting a consistent set of library units for input to the Linker. Only those units actually used shall be included;

5. provide a "stub generation" facility by which missing bodies of Ada subprograms, package and task specifications may be automatically generated as valid Ada source code and included in a consistent set of library units. Such source code shall be the minimum necessary to permit program execution;

6. provide a linker that shall accept linkable Diana program library units and produce near-executable memory images for input to the KAPSE loader (see 3.1.1.1).

3.7.4 Ada Compiler

The Ada compiler shall translate Ada language source text input and process various forms of Diana, including linkable modules. More specifically, the Ada compiler shall:

1. process the complete Ada language as specified in the Ada Reference Manual [G-2];

2. compile Ada source code statements at a minimum rate of 1000 statements per minute, computed by dividing CPU time by the total number of statements in the compilation unit;

3. be designed in terms of relatively independent processing phases. The information interface between phases shall consist of the various forms of the Diana intermediate language. The phases shall comprise a target machine-independent set and a target machine-dependent set. Retargetting shall affect only the middle and back end, and machine dependencies shall be parameterized where possible.

4. report compilation statistics and produce source, object and cross-reference listings;

5. perform error detection and reporting, including classification of errors by severity;

6. perform optimizations for timing and/or space, without changing the functional meaning of a program;

7. identify any necessary limitation on capacities (such as symbols and nesting level).

3.7.5 Text Editor

The text editor shall provide a basic editing facility suitable for editing general text (characters). More specifically the text editor shall:

1. edit all text files including but not limited to Ada source programs;

2. provide sufficient commands to accomplish any arbitrary change, insertion, deletion or rearrangment;

3. be operable in interactive or batch modes;

4. provide an Ada mode in which "words" are defined by Ada lexical rules;

5. accept scripts or files of editing commands saved in the database as control input. Furthermore, the long forms of all commands shall be sufficiently self-explanatory so that scripts are readable by humans as well as the editor;

6. support both standard hard-copy and CRT-type terminals effectively;

7. be able to invoke MAPSE tools from within the editing session. The editor can pass the text it is editing to MAPSE tools as input data, and receive output data from tools into the edited tool.

8. be simple for new users to learn yet satisfying in its capabilities and its interface for most experienced personnel.

3.7.6 Debugging Facilities

The debugging facilities shall be capable of controlling the execution, examination, and modification of an Ada program. It shall process a debug control language that permits a user to specify a variety of actions to be taken at execution control points (breakpoints). (See 3.1.1.6). Such points shall be based on Ada statements or labels, subprogram calls, returns, and exceptions.

When a program in execution is suspended at a breakpoint, a user shall be able to display, dump, or modify selected data values, trace the flow of program control, and modify program instructions. The design shall also provide interfaces for functional simulation; that is, future APSE tools may be added to advance a 'pseudo-clock' (in order to keep track of problem time), activate Ada tasking functions based on that time or other multi-programming conditions, and interface to the KAPSE for other support, such as environment model updating.

The design of the system shall be compatible with a future APSE-resident debug tool (using identical or, at least, similar debug commands) that permits control and debugging of embedded computer software executing on a target machine.

3.7.7 MAPSE Generation and Support

MAPSE Generation and Support (MGS) shall provide the facilities to develop the MAPSE on the IBM370/VM or Perkin/Elmer 8/32 hosts and to aid in its maintenance on these hosts. MGS also shall refer to certain system-wide utilizes that are not specifically part of other functional areas. More specifically, MGS shall:

1. provide a bootstrap compiler so that the MAPSE Ada Compiler can be written in Ada itself. (This is necessary because no Ada compiler currently exists for either the IBM370/VM or the Perkin/Elmer 8/32);

2. provide "secondary" MAPSE tools whose primary function is to build, maintain, and extend the MAPSE itself. Two such tools identified at this level are the lexer and parser generators.

3. provide a virtual memory management system to aid in the definition and access of database objects. This system shall include the capability to define arbitrary data structures and an access facility to automatically address virtual memory far in excess of the real storage available.

3.8 Precedence

The order of precedence places this document first, the Computer Program Product Specifications next, with the test plans last.

4.0 QUALITY ASSURANCE PROVISIONS

4.1 General

Tests and evaluations shall be conducted to verify that the design and performance of the MAPSE shall meet or exceed the requirements specified in Section 3 of this document. A program of testing and validation augmented by analysis shall be conducted to verify compliance with the requirements of this specification and subordinate specifications. MAPSE test objectives shall be implemented in accordance with the Computer Program Development Plan (CPDP) and the Test Plans for each defined CPCI.

4.1.1 Responsibility for Tests

4.1.1.1 MAPSE Test Plans and Procedures

Unless the contract otherwise specifies, the contractor shall be responsible for the performance of all analysis, tests and demonstrations specified herein. All testing shall be witnessed by the Government.

4.1.1.2 Ada Compiler Validation

The Government shall be responsible, with contractor support, for the validation of the IBM 370 and Perkin/Elmer 8/32 Ada compilers. The government shall employ the DoD Ada Compiler Validation Facility (ACVF) for this purpose. Acceptance of the compilers shall be contingent on certification by the Ada Certification Control Board.

4.1.1.3 Independent Testing

The Government shall engage an independent IV & V contractor to conduct independent testing of the MAPSE software. The design contractor shall support this effort.

4.1.2 Special Tests and Examinations (N/A)

4.2 Quality Conformance Verification

Formal Qualification verification shall be conducted (in accordance with the CPDP) to validate that the design and performance of the system satisfy the requirements of paragraph 3.

4.2.1 Software Quality Assurance

The quality assurance provisions defined in this section provide requirements for the performance verification of the MAPSE computer programs. These requirements shall assure that the performance and design requirements defined in paragraph 3 are met. These verifications shall be implemented in accordance with the Government-approved CPDP, test plans and test procedures.

4.2.1.1 Preliminary Qualification Testing

4.1.1.1.1 Component Level Qualification

The objective of component testing is to validate functional units. A component is defined as an Ada subprogram, package, task or subunit, and/or any computer program element that has a logical function. Each component shall compile successfully and shall further be validated by visual inspection of the code. Input/Output tests shall be utilized, as appropriate, to verify proper control flow, data results, etc.

4.2.1.1.2 Integration Level Qualification

The objective of integration testing is to test related groups or strings of components that constitute defined CPCI's. KAPSE and/or vendor supplied operating system functions may be necessary to successfully complete integration level tests. These tests shall verify inter-component communications and control, and parameter passage.

4.2.1.1.3 System Level Qualification

System level tests shall be conducted on groups of CPCI's processing simulated or real operational data. This level of testing requires an almost totally integrated computer program interface with the environment. Such tests shall demonstrate typical scenarios in the development and/or management of Ada software.

4.2.2 Formal Qualification Testing

Formal qualification testing shall be conducted as specified in the CPDP, on each CPCI in order to to verify that all requirements of this specification are met. Successful completion of such tests, plus those listed below, shall constitute acceptance of the MAPSE by the Government.

4.2.2.1 Demonstrate Rehostability

The MAPSE system shall be demonstrated to be rehostable from the IBM 370/VM to the Perkin-Elmer computers at RADC.

4.2.2.2 Demonstrate Compilability

The IBM 370 and Perkin-Elmer Ada compilers that are developed as part of the MAPSE shall be demonstrated to compile a non-trivial Ada program with at least 50 executable Ada source statements in at most 256K bytes of memory. Compilation speed shall be not less than 1000 Ada statements per minute (see 3.2.1.2).

5.0 PREPARATION FOR DELIVERY

The MAPSE software system (and specified operable subset, viz. Ada compiler plus KAPSE) shall be delivered to the Government in the form of magnetic tapes and listings. The particular formats shall be specified by RADC and defined within DI-E-30145.

MISSION
of
Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C^3I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.


DAT
FILM